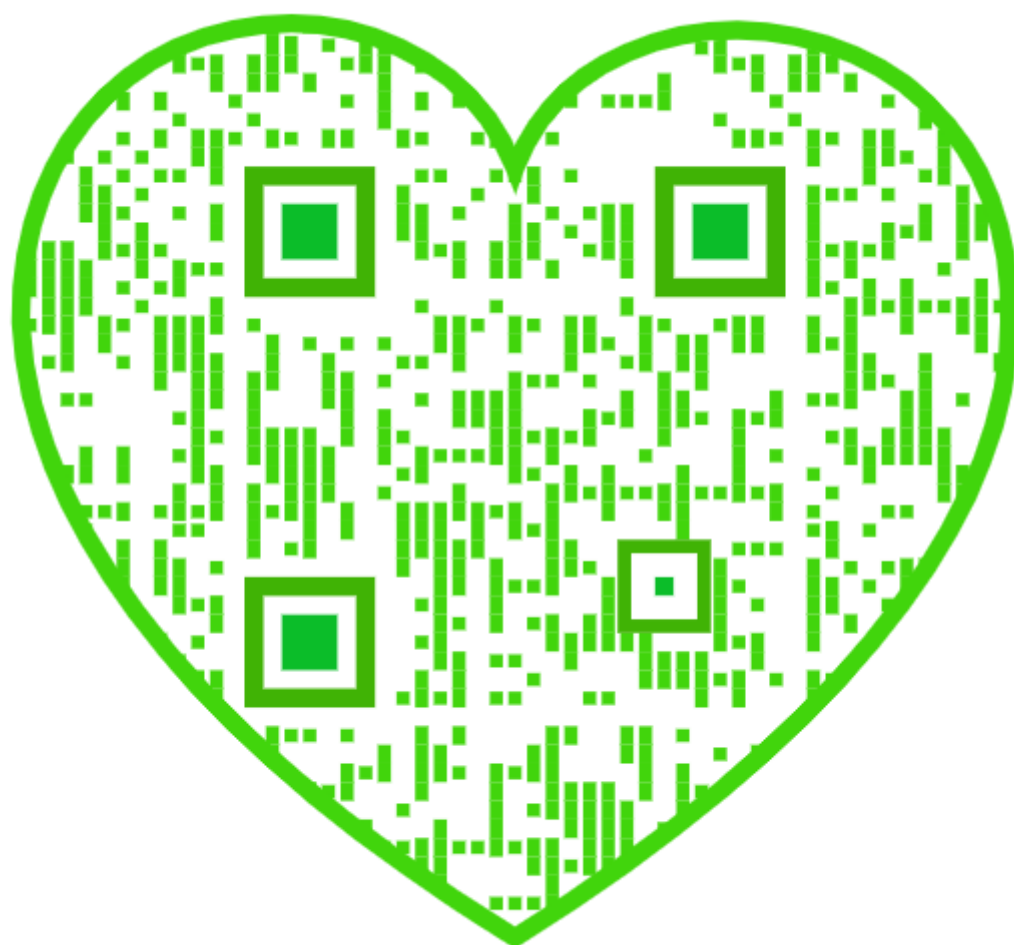


# Master in Artificial Intelligence



## Data Collection & Preprocessing III





# Purpose

**The purpose of the section is to help you learn how to collect and preprocess data to become a Successful Artificial Intelligence (AI) Engineer**

**At the end of this lecture, you will learn the following**

- **How to gather relevant data from various sources, ensure its quality, and preprocess it to make it suitable for analysis and modeling**



# Feature extraction

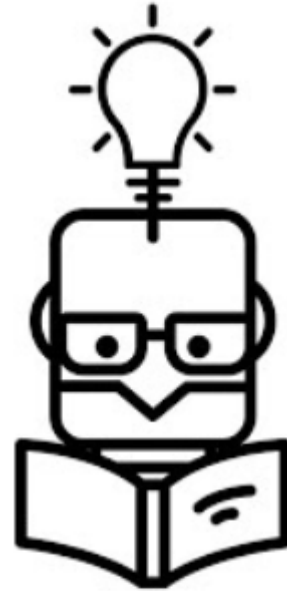
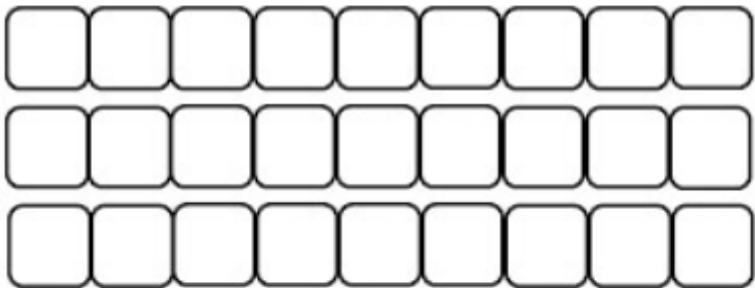


# TF-IDF (Term Frequency-Inverse Document Frequency)

## TF-IDF



$$tf(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$
$$idf(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$



# Calculate Term Frequency (TF)

$$TF(\text{term, document}) = (\text{Number of times term appears in document}) / (\text{Total number of terms in document})$$



# Calculate Inverse Document Frequency (IDF)

$$\text{IDF}(\text{term, collection}) = \log\left(\frac{\text{Total number of documents in collection}}{\text{Number of documents containing term}}\right)$$



# Compute TF-IDF Score

$$\text{TF-IDF}(\text{term}, \text{document}, \text{collection}) = \text{TF}(\text{term}, \text{document}) * \text{IDF}(\text{term}, \text{collection})$$



# Feature Extraction

Document \ Term	information technology	information system	communication technology	software application	telecommunication	computer science
1	0.77	0.55	0.45	0.13	0.14	0.15
2	0	0	0.13	0.53	0.15	0.75

Some Part of TF-IDF Term-Document Matrix



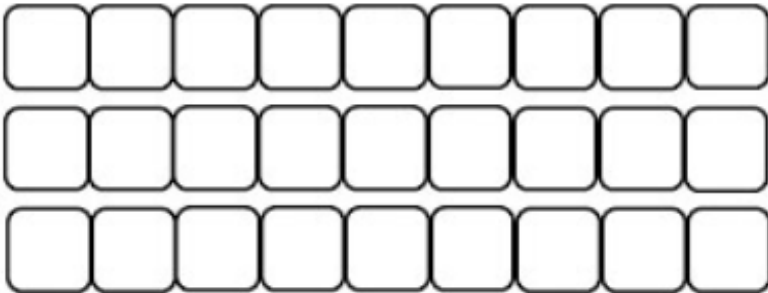
# TF-IDF (Term Frequency-Inverse Document Frequency)

## TF-IDF



$$tf(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

$$idf(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$



# Word embeddings

Suppose we have a small corpus of text documents consisting of three sentences:

1. "The cat sat on the mat."
2. "The dog played in the park."
3. "The bird sang in the tree."

To create word embeddings, we can use the Word2Vec algorithm, which learns distributed representations of words based on their co-occurrence patterns in the corpus. After training the Word2Vec model, each word in the vocabulary is represented by a **dense vector in a continuous vector space**.

Here's a simplified example of word embeddings for the words in our corpus:

- "the": [0.2, -0.4, 0.1]
- "cat": [-0.3, 0.2, -0.5]
- "dog": [0.4, -0.1, 0.3]
- "bird": [0.1, 0.5, -0.2]
- "sat": [-0.2, -0.3, 0.4]
- "played": [0.3, -0.4, -0.1]
- "sang": [-0.1, 0.3, 0.2]
- "on": [0.2, 0.1, -0.3]
- "in": [0.3, -0.2, 0.1]
- "mat": [0.4, 0.2, 0.3]
- "park": [-0.2, 0.3, 0.1]
- "tree": [-0.3, 0.4, -0.2]



# What do the numbers mean in the word embeddings

Suppose we have a small corpus of text documents consisting of three sentences:

1. "The cat sat on the mat."
2. "The dog played in the park."
3. "The bird sang in the tree."

To create word embeddings, we can use the Word2Vec algorithm, which learns distributed representations of words based on their co-occurrence patterns in the corpus. After training the Word2Vec model, each word in the vocabulary is represented by a dense vector in a continuous vector space.

Here's a simplified example of word embeddings for the words in our corpus:

- "the": [0.2, -0.4, 0.1]
- "cat": [-0.3, 0.2, -0.5]
- "dog": [0.4, -0.1, 0.3]
- "bird": [0.1, 0.5, -0.2]
- "sat": [-0.2, -0.3, 0.4]
- "played": [0.3, -0.4, -0.1]
- "sang": [-0.1, 0.3, 0.2]
- "on": [0.2, 0.1, -0.3]
- "in": [0.3, -0.2, 0.1]
- "mat": [0.4, 0.2, 0.3]
- "park": [-0.2, 0.3, 0.1]
- "tree": [-0.3, 0.4, -0.2]



# How are the word embeddings used for Feature Extraction

Suppose we have a small corpus of text documents consisting of three sentences:

1. "The cat sat on the mat."
2. "The dog played in the park."
3. "The bird sang in the tree."

To create word embeddings, we can use the Word2Vec algorithm, which learns distributed representations of words based on their co-occurrence patterns in the corpus. After training the Word2Vec model, each word in the vocabulary is represented by a dense vector in a continuous vector space.

Here's a simplified example of word embeddings for the words in our corpus:

- "the": [0.2, -0.4, 0.1]
- "cat": [-0.3, 0.2, -0.5]
- "dog": [0.4, -0.1, 0.3]
- "bird": [0.1, 0.5, -0.2]
- "sat": [-0.2, -0.3, 0.4]
- "played": [0.3, -0.4, -0.1]
- "sang": [-0.1, 0.3, 0.2]
- "on": [0.2, 0.1, -0.3]
- "in": [0.3, -0.2, 0.1]
- "mat": [0.4, 0.2, 0.3]
- "park": [-0.2, 0.3, 0.1]
- "tree": [-0.3, 0.4, -0.2]



# How are the word embeddings used for Feature Extraction

Sentence Representation



Feature Extraction



Classification Model

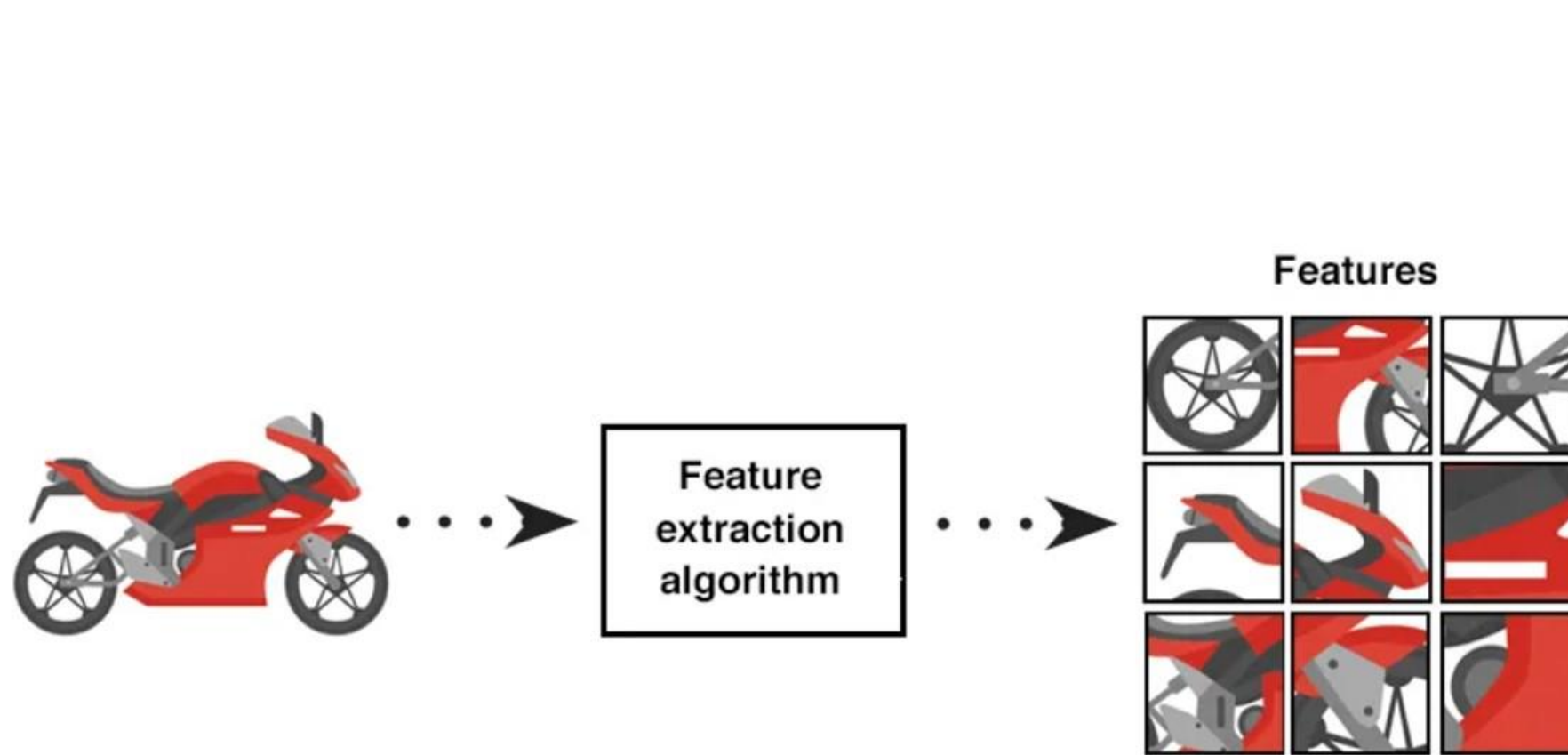


Prediction



# What is next?

## Image feature extraction algorithms



# Master in Artificial Intelligence

*Thank  
you*



## Data Collection & Preprocessing III

